LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-204742

Saving Resources with Plagues in Genetic Algorithms

Francisco Fernandez de Vega, Erick Cantu-Paz, Jose I. Lopez, Tomas Manzano

June 16, 2004

Parallel Problem Solving from Nature
Birmingham, United Kingdom
September 18, 2004 through September 22, 2004

Saving Resources with Plagues in Genetic Algorithms

F. Fernández-de-Vega[1], E. Cantú-Paz[2], J. I. López[1], and T. Manzano[1]

[1] Artificial Evolution Group, Centro Universitario de Mérida,
Universidad de Extremadura. SPAIN
fcofdez@unex.es
[2] Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

Abstract. The population size of genetic algorithms (GAs) affects the quality of the solutions and the time required to find them. While progress has been made in estimating the population sizes required to reach a desired solution quality for certain problems, in practice the sizing of populations is still usually performed by trial and error. These trials might lead to find a population that is large enough to reach a satisfactory solution, but there may still be opportunities to optimize the computational cost by reducing the size of the population. This paper presents a technique called plague that periodically removes a number of individuals from the population as the GA executes. Recently, the usefulness of the plague has been demonstrated for genetic programming. The objective of this paper is to extend the study of plagues to genetic algorithms. We experiment with deceptive trap functions, a tunable difficult problem for GAs, and the experiments show that plagues can save computational time while maintaining solution quality and reliability.

1 Introduction

When researchers apply evolutionary algorithms (EAs), one of the first parameters to be chosen is the size of the population. Usually, difficult problems require a large population, while easier problems can be solved with a small population [1, 2]. A large number of individuals requires a large amount of computing resources, both memory space and computing power for the fitness evaluations. Recently, a technique called plague that dynamically reduces the population size was demonstrated to improve the performance of genetic programming without sacrificing solution quality [3, 4]. In this paper, we demonstrate that plagues can also be successfully applied to genetic algorithms.

Plagues work by periodically removing a number of individuals from the population. Several policies may be employed for selecting individuals to be deleted. For instance, individuals may be removed at random or they can be removed according to some criteria, such as their fitness value, size (in the case of GP), or similarity to other individuals in the population. Plagues can be applied with arbitrary fixed frequencies or they can be triggered by some measure of the progress of the run, such as a measure of population diversity.

In this paper we show, by means of a series of experiments, that plagues may help GAs solve difficult problems in shorter times. The experiments consider a classic tunable problem usually employed to test GAs. As a first study of the application of plagues in GAs, this paper reports results using plagues in every generation, removing a fixed number of individuals per generation, and choosing the worst individuals for removal. We experiment varying the number of individuals removed. Other options of frequency and removal policies are left for future studies. The results suggest that plagues can reduce the computational effort required to reach the global optimum with some certainty.

This paper is structured as follows: section 2 presents a summary of the research related to the plague operator and also some arguments supporting its usefulness. Section 3 describes the way we measure results and the problem employed as a benchmark, while section 4 presents some results obtained using plagues and GAs. Finally we offer our conclusions in section 5.

2 Previous Work

A number of researchers have focused on the study of population size, because of its importance for obtaining solutions of high quality. One of the first studies of population sizing is by Goldberg [6]. His objective was to find the population size that maximizes the rate of schema processing, and found that this rate is maximized with small populations when fitness is evaluated serially and with large populations with parallel evaluations. Goldberg et al. [7] proposed a population sizing estimate based on the variance of fitness. This estimate directly tied population size to solution quality. Goldberg et al. [1] refined the model and obtained a conservative bound on the convergence quality of GAs. Their model is based on the probability of choosing correctly between the best building block and its closest competitor. Later, Harik et al. [8, 2] used some of Goldberg's results to obtain a model that, for some problems, relates accurately the population size to the quality of the solution of a GA.

A few authors have addressed the idea of populations of variable size in GAs. For instance, Kirley applied the concept of "disasters" when using cellular GA [9], where the population is laid on a lattice and individuals interact only with their neighbors. Kirley's idea was to suppress all the individuals in a randomly chosen portion of the lattice. Eventually, the area that suffered the disaster is repopulated with offspring from the individuals that border the disaster area. Since disasters have the effect of fragmenting the population into isolated groups of individuals, it is possible to preserve lesser individuals that would be suppressed quickly in the original population. This mechanism helps maintain diversity in the population that might be useful in later stages of the algorithm.

Smith and Smuda [10] adapted the population size using Goldberg et al.'s population sizing theory [1]. Smith and Smuda employed a dynamic estimate of the variance of building blocks. Tan et al. [11], employ an EA with dynamic population size for discovering the tradeoff surface when solving a multi-objective

optimization problem. They modify the size of populations according to distribution of solutions on the search space.

However, all these proposals differ with the concept of plague described above, because plagues do not take into account the structure of the population (as in Kirley's work) nor the distribution of solutions (as in Smith's and Tan's work).

In genetic programming (GP), people have been even more concerned with deciding the size of populations, because of the well-known problem of bloat [12], which consists of the growth of individuals as the search progresses. Since the total effort depends on the size of the individuals and the population size, deciding the size of the population is important for reducing the computing effort that will be required.

During the last year, the first attempts to directly change the size of the population while the evolutionary process is working were presented in GP. Plagues were first described by Fernández [3] as a new operator acting on the population, and were described in more detail in [5] and [4].

Similar proposals were also described by Monsieurs [13] and Luke [14] later. Luke et al. [14] performed a few experiments to extend the concept of plague to GAs. Their results, however, did not show statistically significant differences in the final fitness values obtained by a fixed-size GA and one with a plague on three optimization problems. Our results differ from Luke's. We not only study mean best fitness values obtained from a set of runs, but also the number of runs that reach the optimum, and these are the curves that better show the advantage of the reduction of population size.

3 Methods

3.1 Computing Effort

In EAs studies, results are commonly presented by comparing a measure of quality (e.g., number of successes in a set of runs, average fitness value) to the number of generations employed. But this comparison is correct only when the computational effort required for computing each of the generations is the same. When conditions change from generation to generation, this way of presenting results is misleading. This is usually the case in GP, when individuals grow as a run progresses. In our case, since plagues reduce the size of populations each generation, the computing power required to evaluate each generation decreases progressively. In [15] and [16] a new way of measuring results under these conditions was presented for GP. Here, we adapt that measure for GA.

We use the *effort of computation* defined as the total number of individuals evaluated for a given number of generations. To calculate this strictly increasing measure, we must first compute the partial effort at a given generation (that we will denote as PE_g) defined as the number of individuals evaluated ($PE_g = i$, i is the number of individuals in the population at generation g). Then, we calculate the effort of computation E_g, at generation g, as

$$E_g = PE_g + PE_{g-1} + PE_{g-2} + \ldots + PE_0.$$

Thus the effort of computation for generation g is the total number of individuals that have been evaluated before generation $g + 1$. Clearly, this measure is problem-specific but it is useful for comparing different solutions and different settings of experiments for the same problem.

It was not our intention here to compare execution times. Our intention is to track the convergence process itself, without directly taking into account actual execution time. However, a reduced effort of computation will imply a reduction in the computation time, regardless of the computer employed.

3.2 Experiments

The experiments used deceptive trap functions, which are used in numerous studies of genetic algorithms because they have known properties and their difficulty can be regulated easily [17]. The values of the deceptive functions depend on the number, u, of bits set to one in their k-bit input substring. The fitness increases with more bits set to zero until it reaches a local optimum, but the global maximum is at the opposite extreme where all the bits in the input are set to one. The order-k traps are defined as

$$f_k(u) = \begin{cases} k - u - d & \text{if } u < k, \\ k & \text{if } u = k, \end{cases} \tag{1}$$

where d is the fitness difference of the two peaks, which in our case is always set to one. The trap functions become more difficult by increasing the number of bits k in the basin of the deceptive optimum and by reducing d.

In the experiments, we varied k from 4 to 8. The fitness functions are formed by concatenating fully-deceptive trap functions and adding their individual contributions. We set the length of the individuals to $l = 20 * k$ bits. For example, for the 6-bit trap problem, the individuals are $l = 120$ bits long and their fitness is calculated as $\sum_{i=0}^{20} f_6(u_{6i})$, where u_{6i} denotes the number of ones in the substring that starts at position $6i$.

We follow previous studies that used deceptive trap functions [2, 19] and set the mutation probability to zero and use pairwise (binary) tournament selection. Although in [19] two-point crossover is used for avoiding excessive disruption on the longer BBs, we used uniform crossover with probability 1. We wanted to make the problem more difficult so that experiments with larger populations are of interest.

Plagues remove a fixed number of individuals every generation. The number of individuals to be removed is a parameter of the operator, and is specified below for each experiment as well as the initial population sizes. When plague is applied, we always remove the worst individuals from the population, according to their fitness value. As we mentioned above, other removal strategies based on fitness or similarity to other individuals are possible, but their study is outside the scope of this paper. Each run was terminated after 50 generations or until the population was empty.

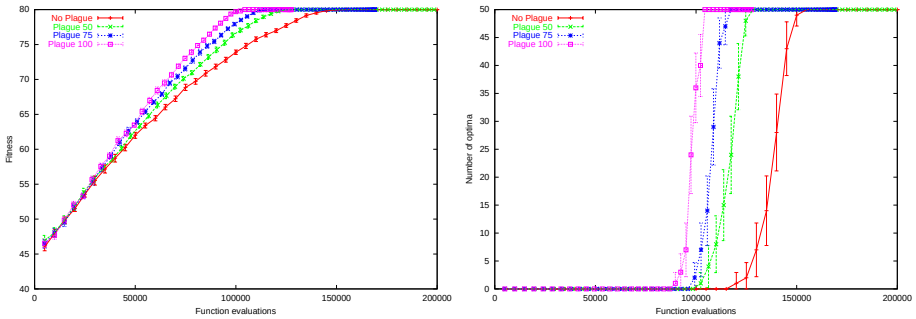


Fig. 1. Fitness (left) and number of times the experiment reached the global optimum out of 50 experiments (right) for different plagues. The initial population size was $N = 5000$ individuals and $k = 4$. The error bars denote 95% confidence intervals.

The experiments consider 50 independent runs for each of the parameter settings. The graphs present the number of times that the GAs reached the global solution or the mean fitness values vs. the number of fitness evaluations.

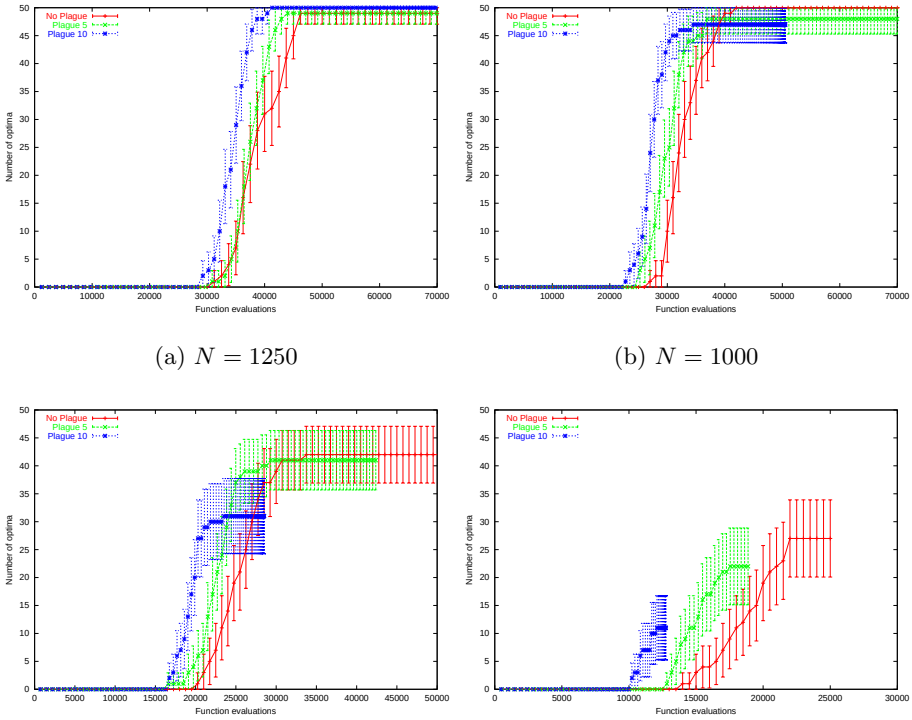
The experiments were carried out using GALib [18]. Only small modifications to the source code were necessary to include the plague operation.

4 Experimental Results

We begun the experiments using the fitness function formed with 20 copies of a $k = 4$ deceptive trap. Figure 1 shows average best fitness vs. the effort of computation and the number of runs that have reached the maximum fitness value for each effort level. In these experiments, the initial population was set to 5000 individuals and the plague was applied every generation removing 50, 75, and 100 individuals. The figure clearly shows that as more individuals are removed, fewer functions evaluations are required to reach a specific fitness level or a number of optima.

The initial population sizes used in the previous experiments are large enough to allow the GAs to find the global solution every time, with and without plagues. This observation suggests that although the plagues saved considerable computational effort, it may be possible to obtain further savings by using smaller initial populations. However, we expect that smaller populations (with or without plague) will result in lower reliability of the GAs and the global optimum will be found less often than with large populations. The next series of experiments examine the effect of plagues on much smaller populations.

Figures 2 and 3 show results obtained when comparing GAs with fixed populations and GAs with plagues. The initial number of individuals in the populations varies from 1250 to 500 individuals, and we experimented with plagues of 5 and 10 individuals. The general trend of these experiments is consistent with the previous experiments with very large initial populations: for a similar effort, the GAs reach the optimum more often with plagues than without plagues and



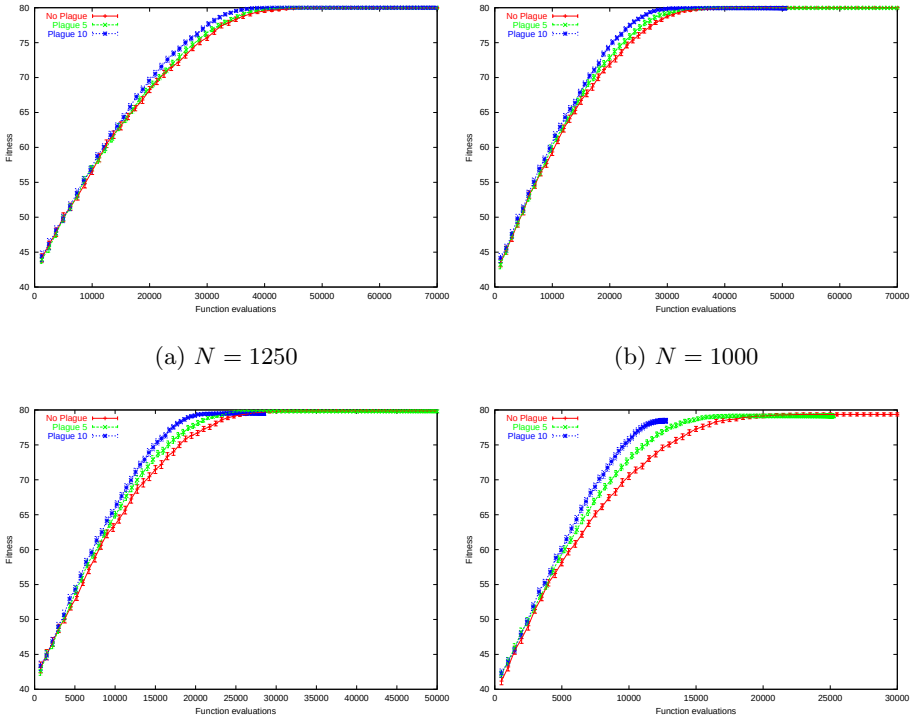
(a) $N = 1250$ (b) $N = 1000$

(c) $N = 750$ (d) $N = 500$

Fig. 2. Number of times that the maximum fitness is reached in 50 runs with and without plagues. In these experiments, $k = 4$ and plagues that remove 5 and 10 individuals per generation were used. Error bars denote 95% confidence intervals.

there are only small differences when the fitness is compared. For $N = 1250$ and $N = 1000$, the plagues save time and maintain the same reliability of the GA without plagues. But as smaller populations are used, the GA has difficulties locating the global optimum consistently. For $N = 1000$, the GA without plagues misses the global about 20% of the time, which is about the same reliability of the GA with a plague that removes 5 individuals. Removing more individuals or using smaller initial populations results in even more failures to locate the global (e.g., see the graph for $N = 500$).

These failures are caused by using populations that are too small to reliably solve the problem in the first place (the GA with $N = 500$ fails about half the time without plagues) and by the plagues emptying these populations quickly. Possible remedies include using smaller plagues or restricting the final population size to a fraction of the original size (instead of zero individuals). In any case, we emphasize that this set of experiments was intended to study when and how the plagues would fail.



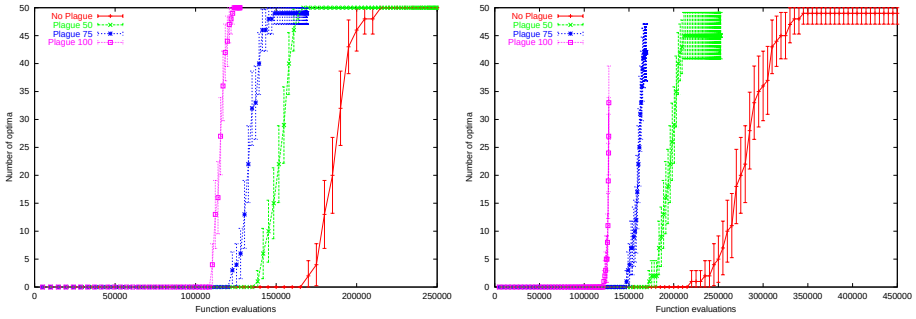
(a) $N = 1250$ (b) $N = 1000$

(c) $N = 750$ (d) $N = 500$

Fig. 3. Mean fitness vs. computational effort for $k = 4$ and plagues that remove 5 and 10 individuals per generation. Error bars denote 95% confidence intervals.

Next, we study how plagues scale up to more difficult problems. We increased the problem difficulty by increasing k to 5 and 6. Given the increase in the difficulty of the problem, the initial size for the population has been also increased to $N = 5000$ individuals. Given the large initial size of the populations, we also enlarged the size of the plagues, removing 50, 75, and 100 individuals per generation. The effect of smaller plagues would be negligible. Figure 4 presents the results of these experiments. As in the case of $k = 4$, removing more individuals seems beneficial. However, the graph for $k = 6$ shows that a few of the experiments did not reach the global optima and that as more individuals are removed, the reliability of the algorithm decreases. This behavior is expected, because as problems become more difficult larger populations are required to solve them and aggressive plagues might delete too many individuals or might delete them too fast (emptying the population before it has enough time to reach the global optimum).

Figure 5 shows results with $k = 8$. We experimented with initial population sizes of 12000 and 14000 individuals and plagues that remove 50 and 75



(a) $k = 5$ (b) $k = 6$

Fig. 4. Mean number of times the experiment reached the global optimum out of 50 experiments. The initial population size was $n = 5000$ individuals. The error bars denote 95% confidence intervals.

individuals per generation. In all cases, the plagues resulted in computational savings.

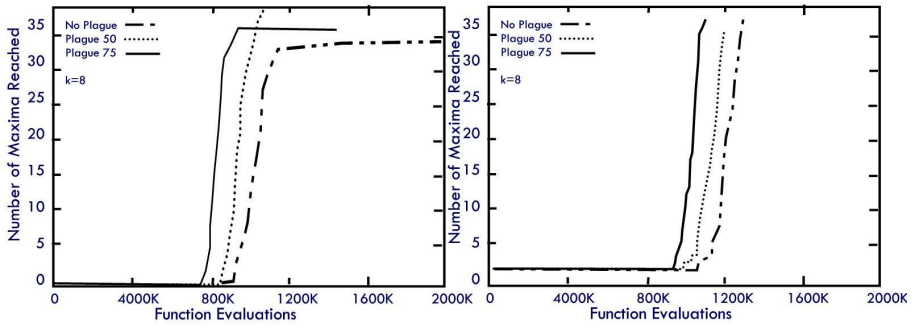


Fig. 5. Number of times that the maximum fitness is reached in 50 runs with and without plagues. $k = 8$.

5 Conclusions

This paper introduced the use of plagues to reduce the population size in genetic algorithms. By means of experiments with a tunable problem, we have shown that plagues help GAs to find solutions of a given quality using smaller efforts than the classic fixed-size algorithm.

Although we have employed only a plain version of the plague that removes a fixed number of the worst individuals in each generation, other elimination

techniques should be tested in the future. Experiments varying the frequency and size of plagues are also necessary to verify the sensitivity of the GA to these parameters.

Acknowledgments

We acknowledge financial support by the Spanish Ministry of Science and Technology (project TRACER) under contract number TIC2002-04498-C05-01. Portions of this work were performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

1. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. Complex Systems **6** (1992) 333–362
2. Harik, G., Cantú-Paz, E., Goldberg, D., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation **7** (1999) 231–253
3. Fernandez, F.: Estudio de poblaciones de tamaño variable en programacion genetica. In: Actas del 2 Congreso Español obre Metaheuristicas, Algoritmos Evolutivos y Bioinspirados, Maeb 03. (2003) 424–428
4. F. Fernandez, M. Tomassini, L.V.: Saving computational effort in genetic programming by means of plagues. In: Proceeding of the Conference on Evolutionary Computation 2003, IEEE Press (2003) 2042–2049
5. Fernandez, F., Vanneschi, L., Tomassini, M.: The effect of plagues in genetic programming:a study of variable-size populations. In Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E., eds.: Proceedings of the Sixth European Conference on Genetic Programming (EuroGP-2003). Volume 2610 of LNCS., Essex, UK, Springer Verlag (2003) 317–326
6. Goldberg, D.E.: Sizing populations for serial and parallel genetic algorithms. In Schaffer, J.D., ed.: International Conference on Genetic Algorithms (ICGA), San Mateo, CA, Morgan Kaufmann (1989) 70–79
7. Goldberg, D.E., Rudnick, M.: Genetic algorithms and the variance of fitness. Complex Systems **5** (1991) 265–278
8. Harik, G., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. In: International Conference on Evolutionary Computation, Piscataway, NJ, IEEE (1997) 7–12
9. Kirley, M., Li, X., Green, D.G.: Investigation of a cellular genetic algorithm that mimics landscape ecology. In McKay, B., Yao, X., Newton, C.S., Kim, J.H., Furuhashi, T., eds.: Proceedings of the 2nd Asia-Pacific Conference on Simulated Evolution and Learning (SEAL-98). Volume 1585 of LNAI., Berlin, Springer (1999) 90–97
10. Smith, R.E., Smuda, E.: Adaptively resizing populations: Algorithm, analysis, and first results. Complex Systems **9** (1995) 47–72
11. Tan, K., Lee, T., Khor, E.: Evolutionary Algorithms with Dynamic Population Size and Local Exploration for Multiobjective Optimization. IEEE Transactions on Evolutionary Computation **5** (2001) 565–588

12. Banzhaf, W., Langdon, W.B.: Some considerations on the reason for bloat. Genetic Programming and Evolvable Machines **3** (2002) 81–91
13. Monsieurs, P., Flerackers, E.: Reducing population size while maintaining diversity. In Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E., eds.: Proceedings of the Sixth European Conference on Genetic Programming (EuroGP-2003). Volume 2610 of LNCS., Essex, UK, Springer Verlag (2003) 142–152
14. Luke, S., Balan, G.C., Panait, L.: Population implosion in genetic programming. In et al., E.C.P., ed.: Genetic and Evolutionary Computation – GECCO-2003. Volume 2724 of LNCS., Berlin, Springer-Verlag (2003) 1729–1739
15. Fernandez de Vega, F.: Distributed Genetic Programming Models with Application to Logic Synthesis on FPGAs. PhD thesis, University of Extremadura (2001)
16. Fernandez, F., Tomassini, M., Vanneschi, L.: An empirical study of multipopulation genetic programming. Genetic Programming and Evolvable Machines **4** (2003) 21–51
17. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In Whitley, L.D., ed.: Foundations of Genetic Algorithms 2, San Mateo, CA, Morgan Kaufmann (1993) 93–108
18. Wall, M.: Galib 2.3.2 (1995)
19. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Accademic Publishers (2000)